Trabajo práctico especial

Programación Orientada a Objetos



Alumno: Luján Nicolás

Email: [Nico.L.2014@hotmail.com](mailto:Nico.L.2014@hotmail.com)

Ayudante designado: Feldman Juan

1)

MultiQueueInterspersed

GetQueue(List<Queue>): Queue

MultiQueueRandom

GetQueue(List<Queue>): Queue

List <Queue> colas

set(Object): void

Get(): Object

AgregarCola(Queue):void

GetCantElementos(): int

GetMaxFifo():QueueFifo

GetCantQueueFifo(): int

Get(): Object

Set(Object):void

GetCantElementos():int

GetMaxFifo():QueueFifo

GetCantQueueFifo():int

Queue(interface)

AbstractMultiQueue

List <Object> elementos

Get():Object

Set(Object):void

GetCantElementos():int

GetMaxFifo():QueueFifo

GetCantQueueFifo():int

QueueFifo

Para plantear la solución del inciso 1 de práctico especial lo que se hizo fué plantear una interfaz Queue dónde sus métodos son abstractos. Posteriormente para agregar la funcionalidad que se pide a la lista de colas de colas se planteó una cola terminal (QueueFifo) encargada de almacenar los elementos que se guardarán en la cola (Almacenando dichos elementos en una lista de objetos). Por otro lado para agregar la funcionalidad de añadir colas de colas se creó la clase abstracta “AbstractMultiQueue” y, además, para agregar la propiedad de sacar y colocar de manera intercalada y random se implementaron dos clases, que heredan de AbstractMultiQueue, cada una con su respectiva funcionalidad.

2) Para poder almacenar un tipo primitivo en la cola lo que se debe hacer es parametrizar la lista de la cola terminal para que pueda almacenar datos de tipo Object, y posteriormente, usar valueof para “transformar” los datos primitivos (int en este caso) a datos de tipo Object (Objeto) para así, de esta forma, poder almacenar enteros a nuestra cola (Todo esto quédo obsoleto en la última actualización de java, ya qué, el “pasaje” de un tipo primitivo a un tipo objeto se hace automáticamente).

3)

PoliticRandom

GetQueue(List):Queue

PoliticInterspersed

GetQueue(List):Queue

MultiQueue

List<Queue>

Politic Politica

Get():Object

Set(Object):Void

GetCantElementos():Int

GetMaxFifo():QueueFifo

GetCantQueueFifo():Int

QueueFifo

List<Object>

Get():Object

Set(Object):Void

GetCantElementos():Int

GetMaxFifo():QueueFifo

GetCantQueueFifo():Int

Queue(interface)

Get(): Object

Set(Object): Void

GetCantElementos():int

GetMaxFifo():QueueFifo

GetCantQueueFifo():int

Politic

getQueue(List):Queue

Para resolver el inciso 3, es decir, para cambiar la forma de colocar y extraer en tiempo de ejecución lo que se hizo fué declarar otra clase que se encargue de dicho funcionamiento. En la Cola múltiple se declaró una variable distancia encargada de manejar la política de la cola múltiple (de colocar y extraer intercalado o random).

4 y 5)

Politic

GetQueue(List<Queue>): Queue

PoliticInterspersed

GetQueue(List<Queue>): Queue

PoliticRandom

GetQueue(List<Queue>): Queue

PoliticMax

GetQueue(List<Queue>): Queue

PoliticMin

GetQueue(List<Queue>): Queue

Queue(interface)

Get(): Object

Set(Object): Void

GetCantElementos(): Int

GetCantQueueFifo(): Int

MultiQueue

List<Queue> elementos

Politic PoliticaExtraer

Politic PoliticaColocar

SetPoliticaExtraer(Politic):Void

SetPoliticaColocar(Politic):Void

Get(): Object

Set(Object): Void

addCola(Queue): Void

GetCantElementos(): Int

GetCantQueueFifo(): Int

addMin(Object): Void

addMax(Object): Void

QueueFifo

List<Object> elementos

Get(): Object

Set(Object): Void

GetCantElementos(): Int

GetCantQueueFifo(): Int

Para resolver el inciso 4 lo que se hizo fué usar la funcionalidad de Politic y declarar dos variables distancias en MultiQueue encargadas de manejar la política de la cola de manera que se pueda cambiar la funcionalidad de colocar o extraer de forma dinámica. Por otro lado para resolver el inciso 5 lo que se hizo fué declarar dos clases mas que heredan de politic y encargadas de extraer la cola máxima o mínima (dependiendo de la cantidad de elementos) para así colocar o extraer en la cola con más o menos elementos.